This repository Search Explore Gist Blog Help joantolos

joantolos / **business**

Unwatch 1 Star 0 Fork 0

Java, Spring MVC, MySQL template. — Edit

46 commits **1** branch **0** releases **1** contributor

branch: **master** **business** / +

proper conf pom

jtolos authored 3 days ago latest commit 0e119f710c

business-api	minor changes	8 days ago
business-backend	pakage renaming	8 days ago
business-batch	new batch module	8 days ago
business-common	test package refactor	8 days ago
business-conf	proper conf pom	3 days ago
business-ear	not useful	8 days ago
business-frontend	pakage renaming	8 days ago
LICENSE	Initial commit	14 days ago
README.md	renaming	11 days ago
pom.xml	proper conf pom	3 days ago

Code
Issues 0
Pull Requests 0
Wiki
Pulse
Graphs
Settings

HTTPS clone URL
https://github.com/:

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop
Download ZIP

README.md

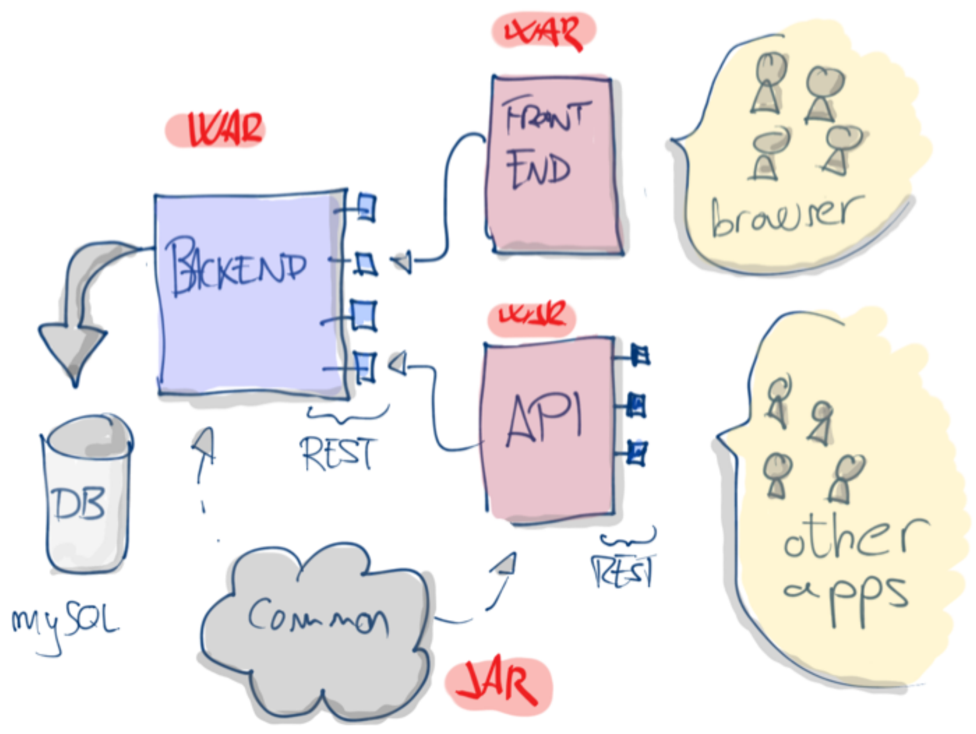
Business Template

Java, Spring MVC template.
Provides the basic functionallity to build a web application with:

- front-end
  - bootstrap
  - javascript
  - json
- back-end
  - json
  - database capabilities: mysql
- configuration files
  - properties files separated by environment

Besides some extra utilities to deal and manipulate Strings, Files and Json.

Architecture



The schema above represents an overview of the general architecture. Very simple design with three war files:

- **BACKEND:** Communicates with the data base and expose the REST services to be consumed by the front-end.
- **FRONTEND:** Consumes the services from Backend and shows the data on a gui based on Bootstrap.
- **API:** Consumes the services from Backend and expose external services for third parties to consume.

And one jar file:

- **COMMON:** Provides common utilities for all the other modules.

Starting the WebApp

You will need Maven to compile the application. Just type

```
mvn clean install
```

on the **'../webapp/'** folder that contains the parent pom. This will compile the frontend, backend, common and api module. You can find the single war file for the frontend, backend and api modules on their respectives target folders, or you can use the ear file generated on the module "webapp-ear".

Deploying

You can deploy the single war files on the same or different machines with tomcat, jboss, etc... or you can use the ear file located on "../webapp/webapp-ear/target" and use tomme or jboss to deploy on the same instance. This is useful for developing on localhost.

Testing the WebApp

Use this url to see if the backend is up:

```
http://localhost:8080/webapp-backend/
```

Use this url to see if the front-end is up:

```
http://localhost:8080/webapp-frontend/
```

Use this url to test the ping-pong service:

```
http://localhost:8080/webapp-frontend/ping
```

You can also click the button on the greetings page to see the Ping Pong service.

Change the default port (8080) if you have some custom configuration on your server.

The API

The API war is located on the target folder of the api module and it is included on the ear file as well.

To access the API rest endpoint you can access the URL:

```
http://localhost:8080/webapp-api/
```

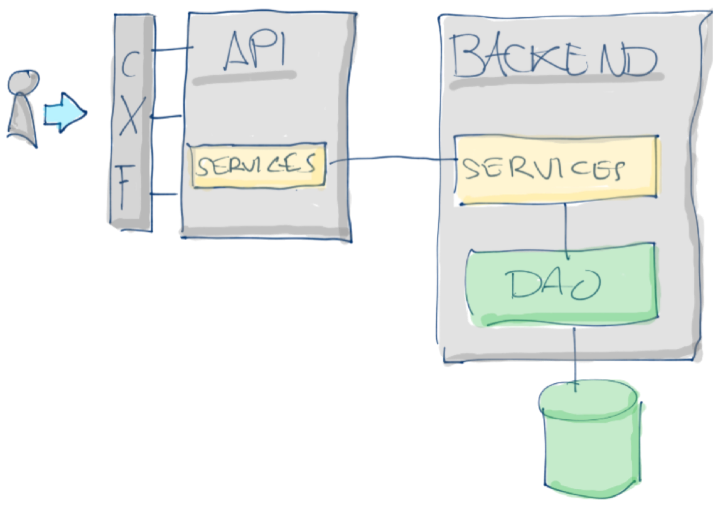
To test the ping-pong service you can try:

```
http://localhost:8080/webapp-api/rest/playPing
```

And it's prepared to receive a parameter (kind of motion):

```
http://localhost:8080/webapp-api/rest/playPing
```

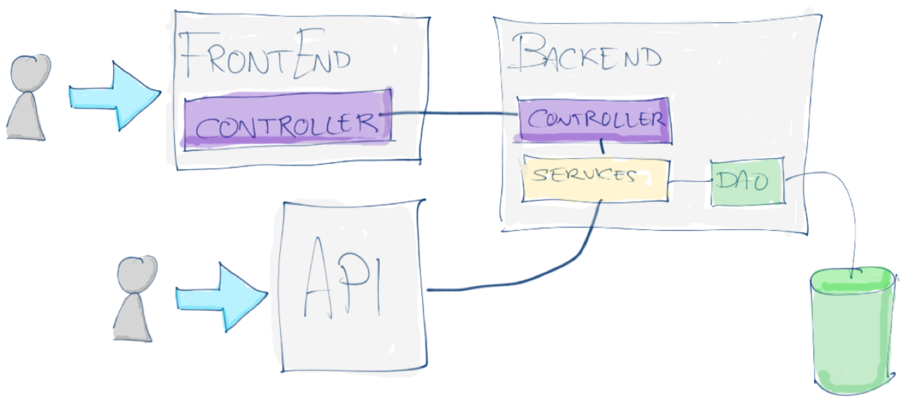
The idea is to provide with REST and SOAP services to third party to build their app on top of the API. Both the API and the frontend use the same services deployed on the backend war.



Start hacking

Five basics points for start editing the code:

- **PingController.java** on the frontend module: You can map the requests with the corresponding service.
- **PingService.java** and **PingServiceImpl.java**: on the frontend module: What the request will do is basically call a service living on the backend module.
- **PongController.java** on the backend module: Receives the request and returns the logic implemented on the corresponding service.
- **PongService.java** and **PongServiceImpl.java** on the backend module: Implements the logic of the service (go to database, etc)
- **business.properties** file where the endpoints url are defined.


FRONTEND
CONTROLLER
BACKEND
CONTROLLER
SERVICES
DAO
API